The Not So Agile Aspects Of Agile Development

By

Derek Scoble, PMP

AssureIT
THE RIGHT DECISION

Agenda

- Generations of Software Development
- Two Primary Development Approaches
- Agile Discussion
  - So what is Agile
  - Why do Agile
  - What few talk about
  - What the experts say
- Bottom-Line

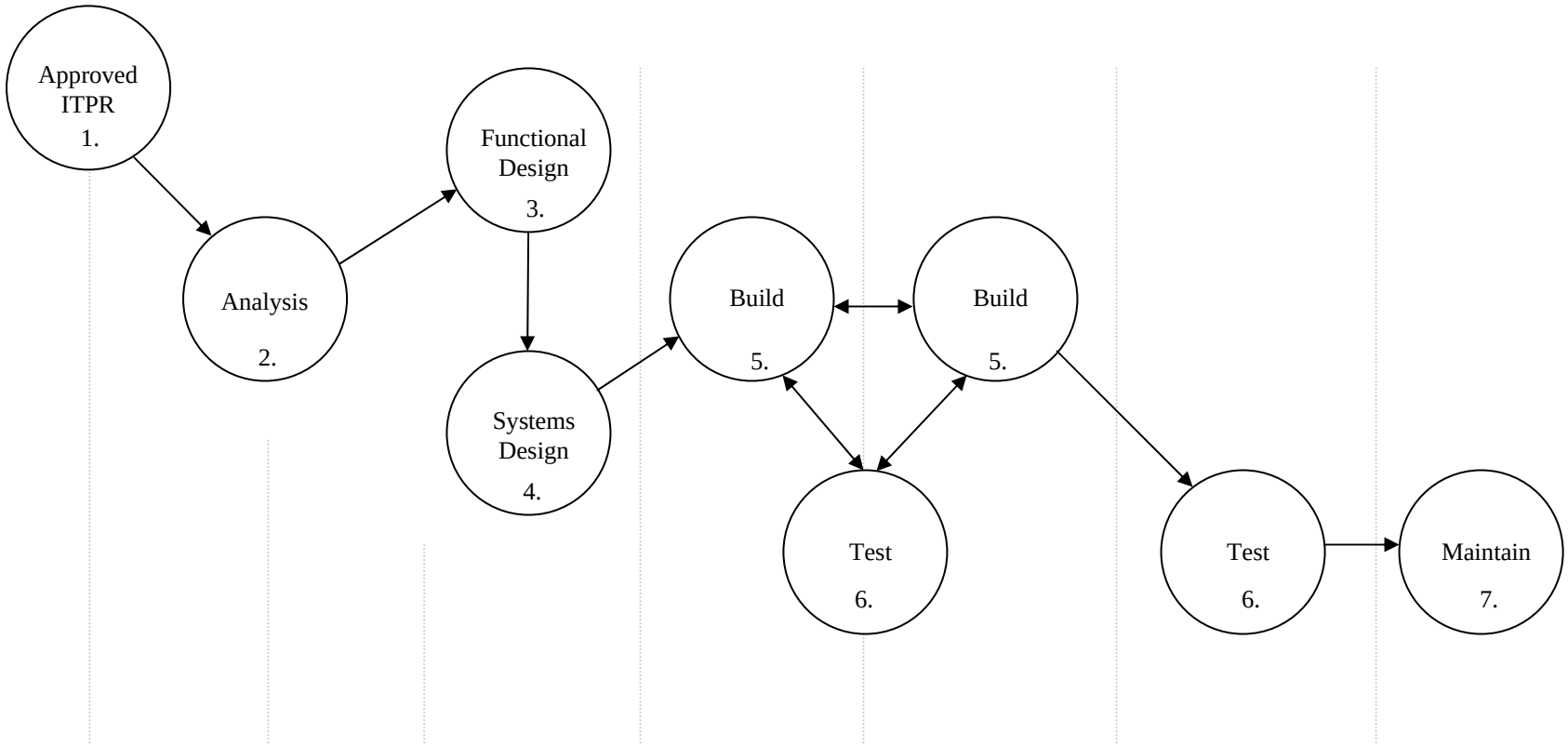
Generations of Development

- There are multiple generations of development techniques available (planned – agile)
  - **1st Generation – Structured Analysis & Design**
  - **2nd Generation – RAD, JAD, Object-Oriented Analysis & Design**
  - **3rd Generation – Agile Development**

Two Primary Development Approaches

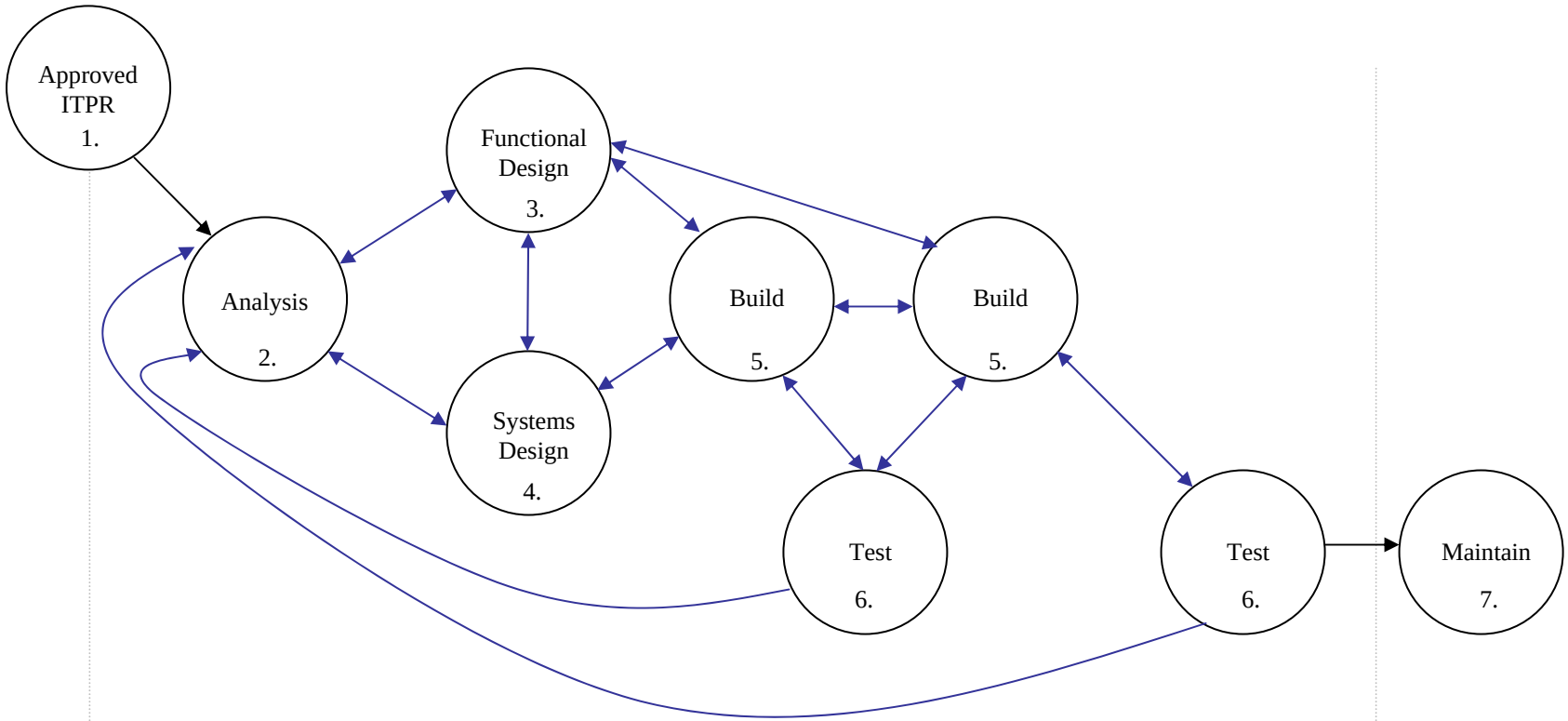
- Planned-driven techniques are considered traditional in nature:
  - **Where defined & repeatable processes support structured approaches**
- Agile-driven techniques are considered evolutionary in nature:
  - **Treats development as a craft, not an industrial process**

Project Example (Planned)



Project Initiation	Planning	Requirements	Functional & System Design	Programming	SW Integration & Testing	Installation, Acceptance, & Closeout	Operations & Maintenance
State of Montana Software Development Life Cycle Phases							

Project Example (Agile)



Project Initiation	Planning	Requirements	Functional & System Design	Programming	SW Integration & Testing	Installation, Acceptance, & Closeout	Operations & Maintenance
State of Montana Software Development Life Cycle Phases							

Development Techniques

- **Segmented (Waterfall) Development:**
  - Where large systems are broken down into functional segments that worked individually as part of an overall project
- **Spiral Development:**
  - Repeats planning, requirements, & design phases that produce prototypes that are evaluated & improved in subsequent cycles

Techniques (Cont.)

- **Rapid Prototyping:**
  - Most important & critical requirements are quickly designed & implemented in a prototype, evaluated, grown or discarded
- **Iterative Technique:**
  - Similar to segmented, related functionality is divided into working components called “drivers” that are progressively elaborated

Techniques (Cont.)

- **Rapid Application Development (RAD):**
  - Development based upon incremental deliveries of working functionality every 3 to 4 months
- **Joint Application Development (JAD):**
  - Is a RAD concept that involves cooperation between developers & end users to jointly design & develop a working system

Techniques (Cont.)

- **Object-Oriented Development:**
  - Development of components that facilitate re-use of code objects that model real-world objects, attributes, relationships, messages, events, & states
- **Agile Development:**
  - Test-driven development using light methods to deliver working code in 2 – 6 week increments (Crystal, XP, Scrum, etc.)

So What is Agile

- **In a nutshell**
  - Fast turn around Iterative & incremental deliveries
  - Emphases on collaboration
  - Continuous improvement
- **Movement formalized with the Agile Manifesto of 2001**
  - **Individuals & interactions** over Process & Tools
  - **Working software** over comprehensive documentation
  - **Customer collaboration** over contract negotiation
  - **Responding to change** over following a plan

Why Do Agile

- Supposedly **(when implemented correctly)** to:
  - Accelerate product delivery times
  - Facilitate business systems change flexibility
  - Potentially lower development costs
- By Addressing Project Killers Such as:
  - Poor communication, incomplete requirements & specifications, scope issues, inadequate testing, & systems integration

Why Do Agile (Cont.)

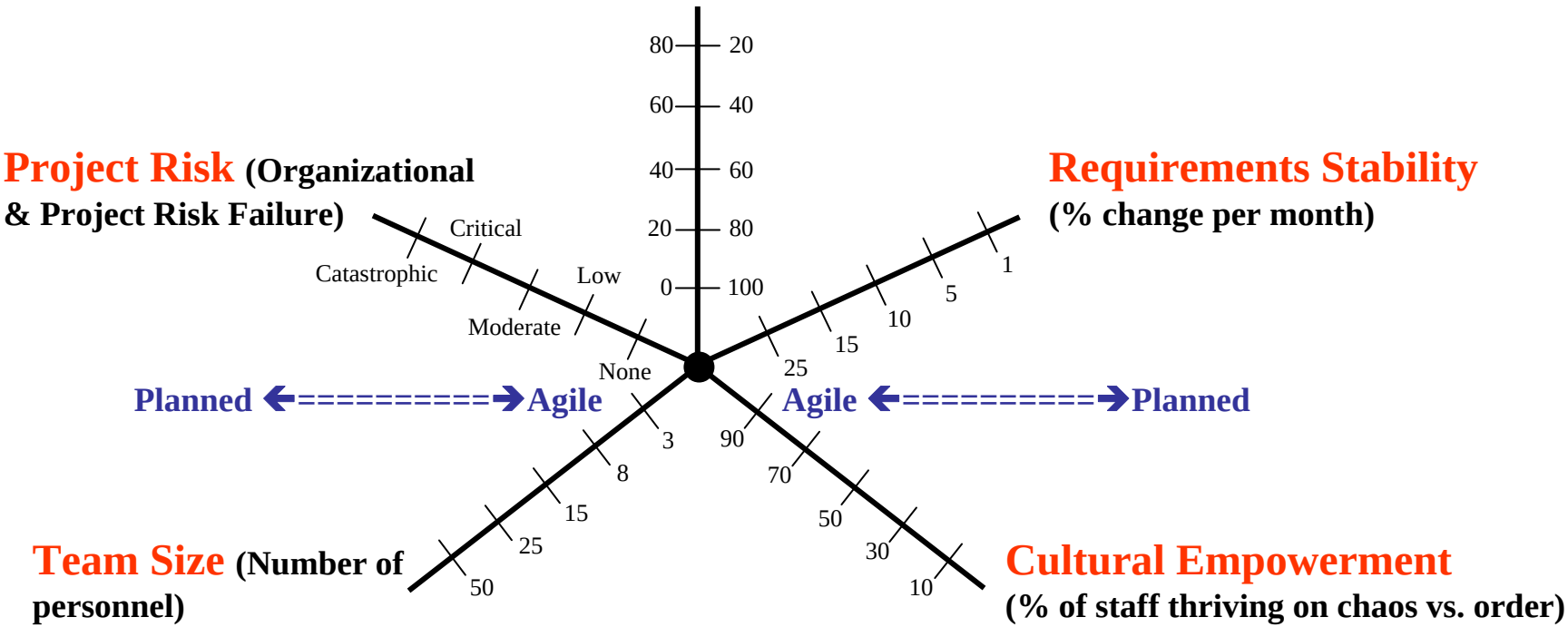
- The primary factors for using **Agile** are responding-to-change driven:
  - **Reduces cost** of moving information between people by placing people together and replacing documentation with talking
  - **Reduces the elapsed time** for decision-making
  - **Smaller increments** (eat the Elephant a bite at a time)
  - **Incrementally test & validate** user expectations

What Few Talk About

- Deciding which technique to use is project & organization, & environment specific:
  - Generally, agile techniques are reserved for **highly skilled teams** with a history of **high performance with superior developmental support tools in a risk-taking culture where schedule & cost risk is accepted on smaller projects (no large body of work demonstrating its scales well - yet)**
  - Generally, planned techniques are reserved for **risk-adverse cultures** utilizing **mixed team skills, less-robust development environments, & larger projects** with **heavier oversight requirements**

Technique Applicability

Staff Skill Ratio's
(% lower skills) – (% of highest skilled staff)



Project Risk (Organizational & Project Risk Failure)

Requirements Stability (% change per month)

Planned ←==========→ Agile **Agile ←==========→ Planned**

Team Size (Number of personnel)

Cultural Empowerment (% of staff thriving on chaos vs. order)

Adapted for Montana from Barry Boehm Balancing Agility & Discipline, 2003

Discipline & Agility at Odds?

- **Discipline** provides strength & comfort
- **Agility** releases & invents
- Structured processes reduce chaos, Agile methods lightens process
- What is perplexing is **finding the balance** to change quickly while not increasing the risks of unknowns – budget, schedule, etc.
- **Projects need both Discipline & Agility**

What the Experts Say

- **Agile is a High-Discipline Methodology** – Alistair Cockburn
- **What's that mean** - Agile development takes professional & proven skills & experience to implement properly. It takes a personal & organizational commitment to foundational aspects of software engineering (requirements, design, test, etc.) time management, planning, estimating, managing commitments & quality
- **Analogy** - Just because I have a drivers license doesn't make me a good driver or qualify me to drive in the Daytona 500

What the Experts Say (Cont.)

- **Agile programming has fallen short** – Steve McConnell
- **What's that mean** – Just like CASE, initial excessive enthusiasm about emerging technologies (Agile) meet reality as the focus shifts from individuals & interactions to process & tools. There is a real difference between academic endeavors & what can be reasonably done
- **Analogy** - We all can strive for new ways of doing things but will the environment support that new direction? Will you get unlimited access to customers as needed? Can you operate with loose budgets & schedules? Do you receive the best training & support tools? Do you have to do other jobs?

10 Common Mistakes in Transitioning to Agile

1. **Go all in**
2. Go *fast* just to go *fast*
3. **Ignore** the corporate culture
4. Fail to **engage** the Sponsor
5. Fail to define **Agile** team roles
6. Fail to do **up-front analysis & monitor continuously** the real-time execution

Adapted from 10 mistakes in Transitioning to Agile, Levent Gurses, Dr. Dobbs Journal 12/06

10 Common Mistakes in Transitioning to Agile (Cont.)

7. **Overdo** the Team-room concept
8. **Trash all** computer-based project management & modeling tools
9. Choose your key resources **poorly**
10. Make **Agile** the new religion

Remember: Sometimes you have to go slow so you can go fast

Adapted from 10 mistakes in Transitioning to Agile, Levent Gurses, Dr. Dobbs Journal 12/06

Bottom-Line

- **The experts mostly agree** that **Agile** methodologies appear to be the future
- **Traditional** techniques place an emphases on process, **Agile** on product
- **Agile**, like traditional development can be either done well or not done well
- **Any development technique requires** supporting investments in people

Q&A